Pico Processor Using Verilog HDL

Vignesh Veerapandian, Xingguo Xiong, Lawrence Hmurcik
Department of Electrical and Computer Engineering
University of Bridgeport, Bridgeport, CT 06604

Abstract

The Pico processor is a scaled down RISC processor hence the name "Pico". Pico processors form an integral part in a network. They act as co-processors to Network processors. The network processors are in-charge of various complex functions such as routing, packet switching, queuing, encryption, decryption, pattern matching, computation and other such tasks. Many Pico processors work in parallel with the network processor, which leads to reduced computing time and improved performance (speed). This in turn increases the processing power of the network processor. One of the main uses of the Pico processor is to take care of the computation part of the network processor. Our project aims to further improve the performance of the network processor by increasing the processing speed of the Pico processor. We can do this by altering the architecture of the current Pico processors to accommodate a five stage pipeline. By doing so, we can manage to increase the speed of execution of each instruction by up to five times. The five stages which we have incorporated in our architecture are Instruction Fetch, Instruction Decode, Execute, Memory I/O and Write Back. The Pico processor is designed and simulated with ModelSim 6.2c. The logic synthesis of the Pico processor is performed using Quartus II software. The simulation results demonstrate the correct functions of the designed Pico processor. Significant performance enhancement has been observed in the designed Pico processor.

I. Introduction

A network processor is an integrated circuit which has a feature set specifically targeted at the networking application domain. Network processors are typically software programmable devices and would have generic characteristics similar to general purpose central processing units that are commonly used in many different types of equipment and products. Pico processors act as co-processors to network processors. A pico processor is an 8-bit processor. It is similar to an 8-bit microprocessor but has an instruction set more similar to a RISC processor. The pico processor has separate instruction and data memories. The instruction memory is 4k instructions in size with each instruction 19-bits wide. The data memory capacity is 256 bytes. The picoprocessor can also address Input and Output devices using up to 256 input ports and 256 output ports. Within the processor there are eight 8-bit general purpose registers, R0 to R7. R0 is always read as zero and ignores writes. There is also return address stack of implementation-defined depth, an interrupt return register and Zero (Z) and Carry (C) condition codes. At present the available pico processors employ two major types of architectures: single cycle unpipelined architecture and multi cycle unpipelined architecture.

Single cycle unpipelined architecture is very basic and simple to design. The execution of an instruction is triggered off by the occurrence of a single clock pulse. The moment a clock pulse occurs the instruction is fetched, decoded and executed. The next instruction is fetched only after the complete execution of the previous instruction. Once it is execute the next instruction is fetched on the next clock pulse.

Multi Cycle Unpipelined Architecture is slightly different from the single cycle architecture. Here also only one instruction is executed at a time, but the difference is that it works on a state machine. Each state

of the processor is triggered by a different clock pulse. Hence, if there are five states then each instruction is executed in five clock cycles. Thus, it is called multi cycle architecture.

In this paper, we presented the implementation of a pipelined pico processor. Due to its pipelined architecture, the speed of processing each instruction is greatly improved. The pico processor which we have developed involves a five stage pipeline. Since there are five stages, each instruction will be executed in one fifth of the time. This clearly affects the overall performance in a positive way. The five stages which we have incorporated also help in reducing the time taken by the processor to execute each instruction.

II. Design of the Pico Processor

The proposed pico processor is based on a pipelined processor architecture. The five stages which we have employed are:

- Instruction Fetch
- Instruction Decode
- Instruction Execute
- Memory I/O
- Write Back

Each stage in the pipeline is given a specific task and it carries it out and transfers the control and the outputs to the next stage. In this way each stage works on the output of the previous stage. The architecture of the pipelined pico processor is shown in Figure 1.

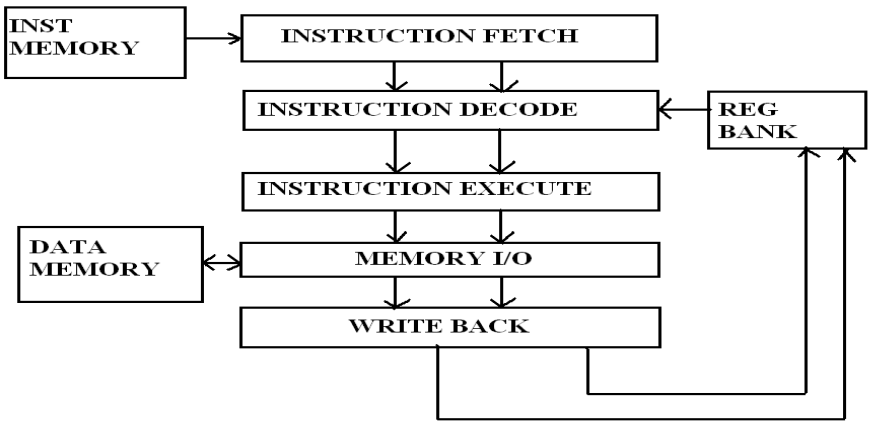

Figure 1. Pipelined Processor Architecture

The working principle of the pico processor can be easily understood with the help of the micro architecture. We have already seen the general structure of a pipelined picoprocessor in the previous chapter. The micro architecture is just the detailed form of the general architecture. It gives us a clear view of how the data moves internally and also tells us how exactly each stage functions. In the micro architecture we are able to see what happens to the instruction and the data in each of the five stages. This helps to understand the internal working of the picoprocessor in a better way. Each stage has many internal and temporary registers. These registers are used for calculations in each stage and to hold the data temporarily till it is passed on to the next stage. The micro architecture of the pico processor is shown in Figure 2.

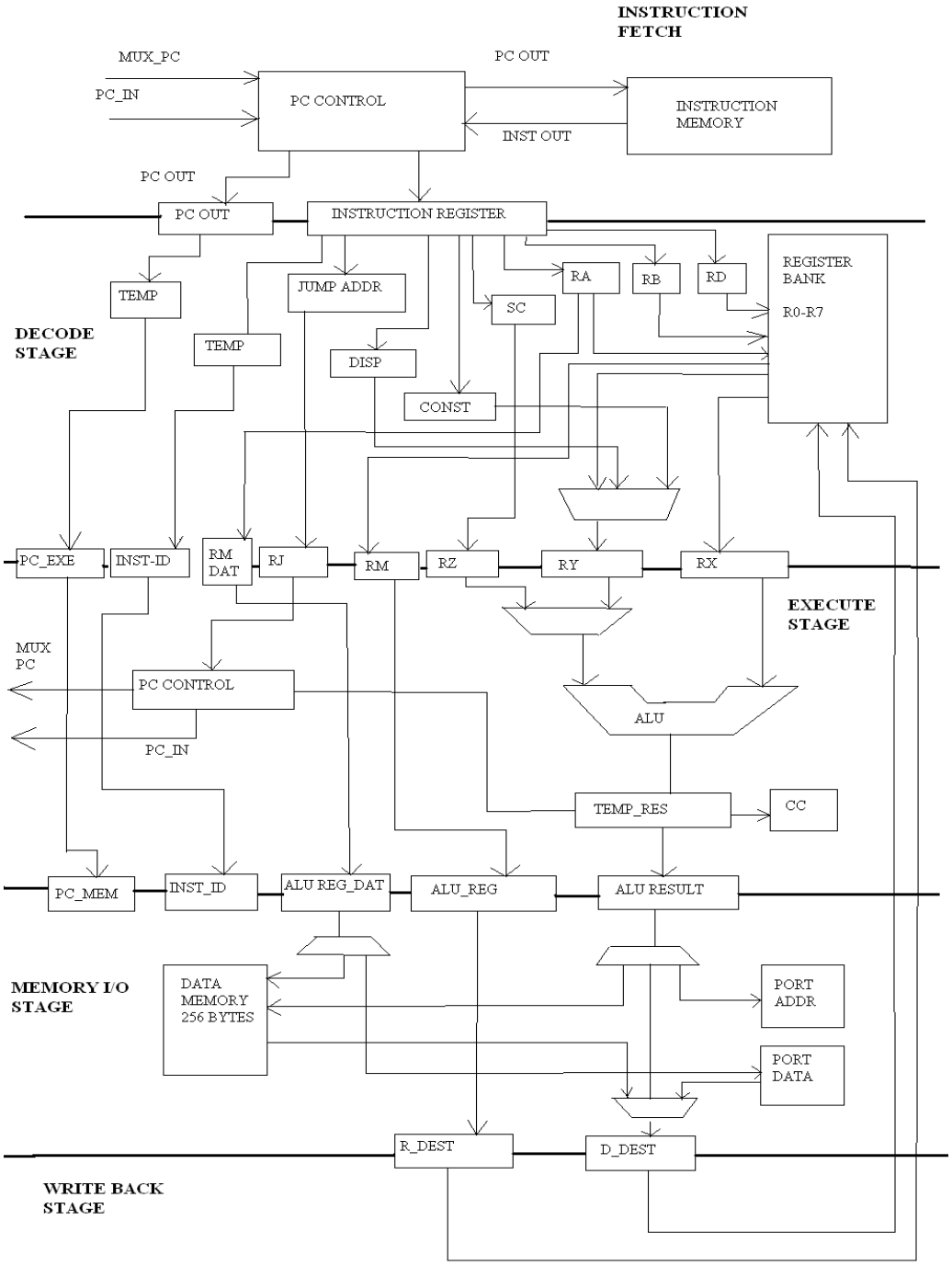

Figure 2. Micro architecture of the pico processor

Now let us see the working of the picoprocessor in detail. For the sake of simplicity let us see each stage separately.

2.1. Instruction Fetch Stage

This is the first stage of the pipeline. The PC control unit decides the value of the PC in order to fetch the instruction from the instruction memory. The MUX_PC signal decides the value of the PC_OUT. If the MUX_PC is low then the PC value is just incremented by 1 and the next instruction is fetched. Otherwise if it is high then the next PC value is taken from the bus and the corresponding instruction is fetched.

Once the instruction is received from the instruction memory it is stored temporarily and then passed on to the next stage along with the current PC value. The PC value is passed for reference. The PC_IN value goes to the PC under special cases like interrupt, jump, branch, ret and reti instructions.

2.2. Instruction Decode Stage

This is the second stage of the pipeline. In this stage the decoding of the instruction. This stage checks the type of instruction and assigns the necessary values to the internal registers. The registers RA, RB, RD hold the 3 bit regbank addresses of the source1, source2 and destination operands. RJ holds the jump address. SC holds the shift count. Registers DISP and CONST hold the 8-bit displacement and constant values respectively.

The three bit addresses are sent to the register bank and the data is received on the two address buses. Based on the type of instruction the data is sent to registers RX and RY. The SC is sent to RZ. The value in RY can be either disp or the const. The RM data contains value which has to be stored in the data memory or sent out on the port. On the occurrence of the clock pulse all the values are passed on to the pipeline registers.

2.3. Instruction Execute Stage

This is the third stage of the pipeline. In this stage the value from RX and either RY or RZ are taken in to the ALU. The value which is computed is stored in a temporary register. The Zero and Carry flags are set based on the ALU result.

The ALU is also used for calculating the effective address in case of memory or input/output instructions. A separate PC control is used to calculate the next value of the PC. In cases like jump, ret, reti, branch and interrupt the PC value is calculated and sent on the PC_IN bus and the MUX_PC signal is made high. In all the other cases the MUX_PC is maintained low and no value is sent on PC_IN.

Once again at the occurrence of the clock pulse the all the temporary values in the internal registers are passed on to the pipeline registers in order to trigger the next stage.

2.4 Memory I/O Stage

This is the fourth stage of the pipeline. In this stage one of the four things happen. Data is either written to the data memory or data is read from the data memory. Then data is either written on the output port or data is read from the input port.

For writing to the data memory or the output port, the effective address of the memory or the port is taken from the ALU result and the data to written is taken from the ALU reg data.

For reading from the data memory or from the input port, the effective address of the data memory or the port is again taken Rom the ALU result. And the destination register address is taken from the ALU reg.

Incase the instruction is not a memory i/o instruction, then at the occurrence of the clock pulse the destination register address and the destination data are written to the registers R_DEST and D_DEST respectively.

2.5. Write Back Stage

This is the fifth and the final stage of the pipeline. Nothing much happens in this stage. At the occurrence of the clock pulse the data in the D_DEST is written to the register bank in the address pointed by R_DEST.

III. Results and Discussion

The Pico processor is designed and simulated with ModelSim 6.2c. The logic synthesis of the Pico processor is performed using Quartus II software. The RTL level netlist of the design is achieved successfully. The simulation results demonstrate the correct functions of the designed Pico processor. Significant performance enhancement has been observed in the designed Pico processor. The screen shot of the top level architecture of the pico processor achieved by logic synthesis is shown in Figure 3. The screen shot of the ModelSim simulation waveforms of the pico processor in execute stage is shown in Figure 4. Due to page limit, the screen shots of the detailed synthesized modules and the simulated waveforms for other stages are not attached.

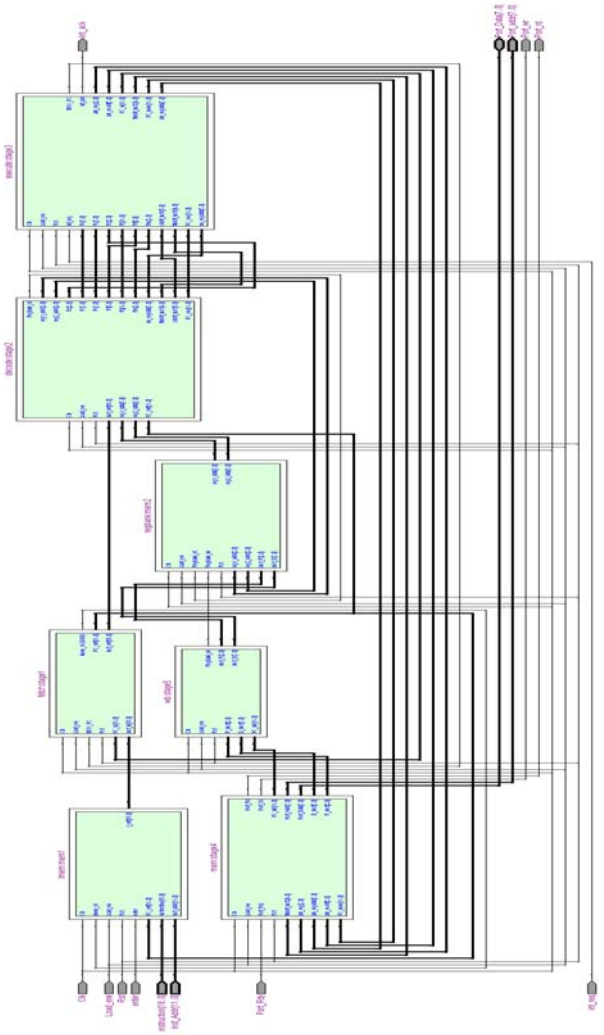

Figure 3. Top level architecture of the pico processor after logic synthsis

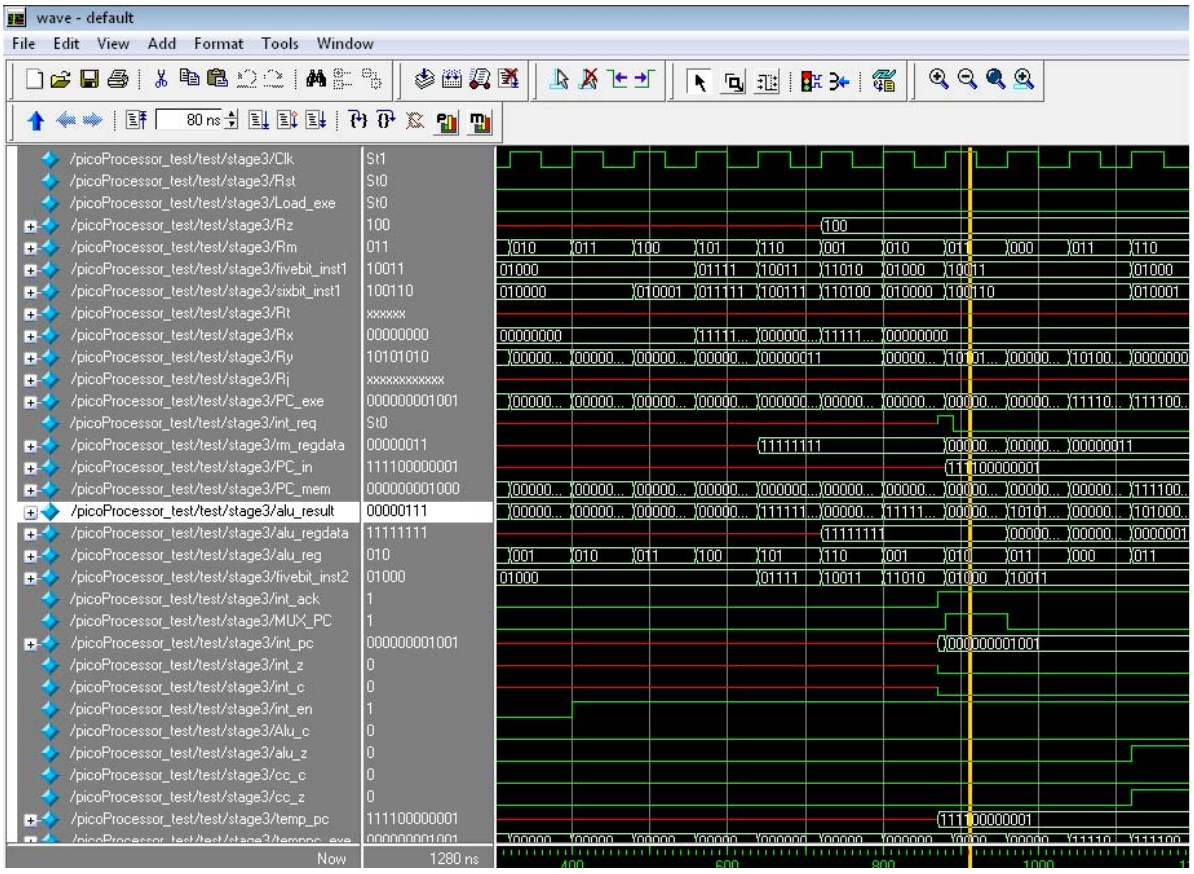

Figure 4. Screen shot of ModelSim simulation waveforms of the pico processor in execute stage

IV. Conclusions and Future Work

In this paper the design and implementation of a pipelined pico processor with Verilog HDL is reported. The five stages incorporated in the pipelined architecture include Instruction Fetch, Instruction Decode, Execute, Memory I/O and Write Back. Due to its five-stage pipelined architecture, the execution speed of each instruction is improved by up to five times. The Pico processor is designed and simulated with ModelSim successfully. The Verilog of the design is further synthesized with Quartus II. The simulation results demonstrate the correct functions of the designed Pico processor. Significant performance enhancement has been observed in the designed Pico processor.

In the future, this picoprocessor can be further developed by increasing the number of pipelined stages, or by multi-threading, parallel execution. The 8-bit processor can be further developed into a 16-bit or a 32-bit picoprocessor. This way the capacity of the picoprocessor will increase and further instructions can be added to the instruction set architecture of the picoprocessor. Adding more pipelining stages will speed up the execution rate of the picoprocessor, which will enhance the network processor to work more quickly and efficiently.

References

[1] David A Patterson and John L Hennessy, *Computer Organisation and Design: The Hardware/Software Interface*, Morgan Kaufmann, 1994, ISBN: 155860281X.

[2] Kai Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, McGraw-Hill, 1993, ISBN: 0071133429.
[3] Carl V Hamacher, Zvonko G Vranesic and Safwat G Zaky, *Computer Organization*, McGraw-Hill, 1996, ISBN: 0071143238.
[4] David R. Smith, Paul D. Franzon, *Verilog Styles for Synthesis of Digital Systems*, Prentice Hall, May 18, 2000, ISBN: 0201618605.

Biographies

Vignesh Veerapandian is a graduate student in Department of Electrical Engineering, University of Bridgeport, Bridgeport, CT. He received his Bachelor degree in Electrical and Electronics Engineering from Anna University, India in 2007. His current research area includes System on chip, Processor and Low Power VLSI.

Dr. Xingguo Xiong is an assistant professor in Department of Electrical and Computer Engineering, University of Bridgeport, CT. He received his Ph.D degree in electrical engineering from Shanghai Institute of Microsystem and Information Technology, China, in 1999. He received his second Ph.D degree in computer engineering from University of Cincinnati, OH, USA in 2005. His research interests include microelectromechanical system (MEMS), nanotechnology, as well as VLSI design and testing.

Dr. Lawrence Hmurcik is the chairman and professor of Electrical Engineering department, as well as professor of Computer Engineering in University of Bridgeport, CT. His research interests include medical electronics, electronic materials and devices, electric circuit simulation, superconductors and various other electrical engineering fields.